IDA PAPER P-2311

MERGING THE Ada COMPILER EVALUATION CAPABILITY (ACEC) AND THE Ada EVALUATION SYSTEM (AES)

Richard P. Morton
Jonathan D. Wood
Audrey A. Hook

October 1989

Prepared for
Ada Joint Program Office (AJPO)

IDA
INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts (a) to record substantive work done in *quick reaction studies*, *(b) to record the proceedings of conferences and meetings*, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract MDA 903 89 C 0003 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

This Paper has been reviewed by IDA to assure that it meets the high standards of thoroughness, objectivity, and appropriate analytical methodology and that the results, conclusions and recommendations are properly supported by the material presented.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
September 1989

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Merging the Ada Compiler Evaluation Capability (ACEC) and the Ada Evaluation System (AES)

5. FUNDING NUMBERS
MDA 903 89 C 0003
T-D5-304

6. AUTHOR(S)
Richard P. Morton, Jonathan D. Wood, Audrey A. Hook

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Paper P-2311

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Ada Joint Program Office
Room 3E114, The Pentagon
Washington, D.C. 20301-3081

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, unlimited distribution.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT *(Maximum 200 words)*

This IDA Paper documents the results of a special analysis requested by the Ada Joint Program Office. The purpose of this analysis was to determine the feasiblity and desirability of merging two separately developed software systems which can be used to expose compiler performance characteristics. The two systems were the Ada Compiler Evaluation Capability (ACEC) and the Ada Evaluation System (AES). The study itself consisted primarily of reviewing documents related to each system. In addition, it was necessary to load both systems and inspect test cases to determine the degree of overlap.

14. SUBJECT TERMS
Ada Programming Language; Ada Compiler Evaluation Capability (ACEC); Ada Evaluation System (AES); Compilers; Test Suites.

15. NUMBER OF PAGES
70

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
UL

IDA PAPER P-2311

MERGING THE Ada COMPILER EVALUATION CAPABILITY (ACEC) AND THE Ada EVALUATION SYSTEM (AES)

Richard P. Morton
Jonathan D. Wood
Audrey A. Hook

October 1989

IDA
INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 89 C 0003
Task T-D5-304

PREFACE

The purpose of IDA Paper P-2311, *Merging the Ada Compiler Evaluation Capability (ACEC) and the Ada Evaluation System (AES)*, is to communicate the results of a special analysis requested by the Ada Joint Program Office (AJPO). The purpose of this analysis was to determine the feasibility and desirability of merging two separately developed software systems which can be used to expose compiler performance characteristics.

The importance of this document is based on fulfilling the objective of Task order T-D5-306, Ada Technology Insertion, which is not a specific deliverable under this task but is a paper mutually agreed upon by the Sponsor and IDA. P-2311 documents the comparison of the capabilities of the ACEC and AES and the resulting conclusions on the desirability and effort for undertaking the merger of these two software software systems. The paper is directed towards the AJPO staff who will make program decisions on the use of compiler evaluation technology.

This document was reviewed on September 18, 1989 by the following members of the CSED Peer Review: Dr. David Carney, and David Hough. An external review was also performed by Dr. John Solomond and Dr. Erhard Ploedereder.

CONTENTS

LIST OF TABLES

MERGING THE ACEC AND THE AES

1. Introduction

1.1 Background

The Computer and Software Engineering Division (CSED) of the Institute for Defense Analyses (IDA) was requested by the Director, Ada Joint Program Office (AJPO) to investigate the technical feasibility of merging the Ada Compiler Evaluation Capability (ACEC) suite of tests developed by the US Air Force and the Ada Evaluation System (AES) developed by the Ministry of Defense (MOD) United Kingdom (UK).

The AES was provided to the US under the terms of the Memorandum of Understanding (MOU) which established the NATO Special Working Group (SWG) on APSE project. The MOU has the following ten nations as current signatories (Belgium, an original signer, has since withdrawn): Canada, Denmark, France, Germany, Italy, Netherlands, Norway, Spain, United Kingdom and United States. Under the terms of the agreement, each of the nations, except France and Denmark, are contributing a part of the APSE or related technology to all the member nations. However, these contributions are restricted outside their developing country to military purposes only. Furthermore, at the moment, there is no agreement on any follow-on maintenance on any contributed product. No new updates to the AES are expected to be received until the NATO SWG on APSE evaluations are performed in approximately two to three years.

The ACEC is being developed by Boeing under contract to US Air Force Systems Command, Aeronautical Systems Division, Wright-Patterson Air Force Base (WPAFB). It is distributed by the Data and Analysis Center for Software (DACS), Griffiss AFB. The ACEC is subject to export control restrictions, and is not available outside the US. Within the US it is available only with the proper approvals. Since the DACS was in the process of changing operating contractors at the time of this study, a copy of the ACEC was borrowed from the AJPO after the specified documents were filed with WPAFB.

The ACEC is designed as a collection of test programs and support packages which are compiled and linked to set up file directories so that the entire collection of test programs can be run after a command script or files have been prepared by the user. The ACEC also has the capability of comparing results from a system under test with results from other systems and computing a normalized statistical comparison.

AES is designed with a test harness and a results database. The test harness functionality allows a user to run one or many test programs, to associate tests into ordered groups, to track the status of tests, to define relationships between tests, and to generate reports where whole phrases or sentences are determined by values in a central results database. The results data base contains all known information about the test suite and results, including the value of implementation dependent information.

1.2 Scope

The authors of this study have interpreted the concept of technical feasibility as encompassing a concern for reasonable benefit if the merge were to take place. Consequently, the study attempted to answer the following questions:

a. What is the benefit to the user in using both products?

b. What is the benefit in combining them?

c. Is there a technically reasonable way to do the merge?

The materials available to the authors were not necessarily the latest products of either supplier of evaluation technology. In particular, the analysis of the ACEC was based entirely on Version 1. We were told that a second version is under development but we did not receive any documentation. Also, another version of the AES, superior to the version available, was demonstrated to one of the team members by the UK MOD. At this time, it is not known if that version will ever be released to the US.

1.3 Organization of this Report

Section 2 of this report describes the approach taken in conducting this analysis. Section 3 contains findings, the facts uncovered in the investigation. Section 4 reports the conclusion which represent our interpretations derived from those findings. Section 5 contains our recommendation for acting upon the conclusions. A bibliography of the documents used for this study is included at the end of the report.

Appendix A presents the analysis of the test suites that lead to Finding 1 is Section 3.1. Appendix B demonstrates the process of converting ACEC tests to run under the AES.

1.4 Acronyms

ACEC	Ada Compiler Evaluation Capability
AES	Ada Evaluation System
AJPO	Ada Joint Program Office
APSE	Ada Programming Support Environment
CSED	Computer and Software Engineering Division
DACS	Data and Analysis Center for Software
DES	Data Encryption Algorithm
EW	Electronic Warfare
IDA	Institute for Defense Analyses
IO	Input/Output
LOC	Lines of Code
MOD	Ministry of Defense
MOU	Memorandum of Understanding
NATO	North Atlantic Treaty Organization
SEI	Software Engineering Institute

SWG	Special Working Groups
UK	United Kingdom
WPAFB	Wright-Patterson Air Force Base

2. Study Approach

Some of the members of the study team had prior exposure to the AES. One member, Jon Wood, had previously installed and used the AES, including writing additional tests for it. Audrey Hook had attended a briefing and demonstration of the AES given by the the UK sponsors.

The study itself consisted of reviewing the documents related to each system, loading both systems, and inspecting test cases. In addition, telephone conversations were held with 1Lt Robert Marmelstein, the ACEC project manager for the US Air Force, and Dr. Nelson Weiderman who has conducted related studies at the Software Engineering Institute (SEI).

3. Findings

3.1 Finding 1: The Overlap Between the Two Test Suites is Small

The data to support this finding is presented in Appendix A. Briefly, the AES comprises 521 tests and the ACEC 1069 tests. Of these, only 53 AES tests appear to be :uplicated in the ACEC. This represents approximately 10% of the AES tests and 5% of the ACEC. However, because of the difference in test generation, it is difficult to count the tests precisely. For example, an AES test file that generates multiple versions of the same test is counted as only one test, but where the ACEC has multiple versions of the same test that differ only slightly, each version is counted.

3.2 Finding 2: The User Interfaces to the Two Systems are Different

The ACEC is completely batch oriented. The tests are executed with the aid of batch command files, and the results are written to files. Two data reduction programs are provided for generating output, one for formating and printing the data produced by the tests, and one for performing statistical analyses on that data and data from comparable tests of other compilers and system configurations.

The AES uses both interactive and batch processing modes. Testing is performed interactively, while reports of results are generated in batch mode. The outputs of the tests are stored in a database, but no user language is provided for retrieving individual or aggregate results interactively.

3.3 Finding 3: Merging the AES into the ACEC Would be Difficult

The primary reason for this finding is that the AES test harness provides functionality that is not in the spirit of the ACEC. This problem is illustrated by two particular areas, capacity tests and the error analysis tests. The tests for the capacity of both the compiler and the run-time environment make use of the capability to dynamically generate tests to conform to the search strategy chosen to determine capacity. One search strategy used in some of the tests is the binary search. In some cases, the test harness asks the user for the initial values of the parameters to be used. Combining both of these capabilities results in a unique series of tests to be run for the specific settings of the parameters chosen. The ACEC has no capability to generate tests dynamically. In effect, the most sophisticated capabilities of the AES test harness would have to be added to the ACEC.

The ACEC currently has no support for tests that fail, but the AES has many tests that are designed to test the robustness of error handling and the readability of error messages by creating situations that are expected to fail. The ACEC functionality would have to be extended to add the capability to capture, analyze and report failure conditions.

3.4 Finding 4: Merging the ACEC into the AES Would be Much Less Difficult

The only functionality of the ACEC that is not specifically in the AES is the Median program. However, since the AES has a statistical analysis package, the major requirement would be to adapt the program to the AES database as its source of input.

Adapting the functionality of one system to the other is only part of the problem. It is also necessary to convert the testˢ from one format to the other. In both cases, the effort required is considerable but straightforward. All of the tests need to be converted in essentially the same way. In converting ACEC tests to AES tests, for example, the statements that write to the output file need to be changed to update the database. The opposite is true for converting from AES to ACEC. The work is seen as tedious but simple, once a design has been established on how one system is to be reflected in the other system.

3.5 Finding 5: The Two Systems Appear to Have Been Developed with Different Primary Users in Mind

Ada compiler and APSE evaluation technology could reasonably be used for several purposes:

1. The selection of a suitable compiling system or tools for a specific project by project managers.
2. Enhancing the understanding of the pluses and minuses of a compiling system or tool by its users (programmers and others).
3. Identifying weaknesses in a compiling system or tool under development by the developing organization.

Generally, managers selecting tools for a project do not want to run the tests themselves. In all likelihood, they would be satisfied with buying the results from an independent and reliable testing organization because that would undoubtedly be the least expensive way to obtain the results, which are only needed once for an acquisition decision.

Users and developers, however, are more likely to be in the situation of wanting to run specific tests many times, possibly even writing some additional tests for some special need. Such users would undoubtedly find the delays of using a testing service to be too slow and costly. They would be much better off having their own copy of the testing system to use whenever it is needed.

This line of reasoning leads to the observation that testing technology is needed to serve the needs of both individual users who want to run their own tests and testing laboratories who need to generate reports suitable for reading by someone else.

Both the ACEC and the AES could be used by either kind of organization, but it appears that each has only one in mind as the primary user. The AES includes a large number of tests of human factors in the form of checklists with supporting test programs. The results are then entered into the database for use in generating reports. The report generator has extensive capabilities for generating English text as part of the reports. These capabilities are most suitable for a testing laboratory whose reports are to be read by others.

The ACEC has no human factors tests. In fact, the ACEC documentation specifically says that there is no need for such tests because just running the test suite will give the user enough exposure to the compiler under test to draw his own conclusions regarding its ease of use. The clear intention here is that the user is the person directly concerned with such issues. The AES contains a large number of tests that require result evaluations involving considerable subjectivity of the evaluator. Examples are AES Group B, C, D, and E tests.

4. Conclusions

4.1 Conclusion 1: There is Benefit in Using Both Test Suites

This conclusion stems from the finding that the overlap between the two test suites is small. The two systems have complementary capabilities: the ACEC tests provide analysis of performance-related criteria related to Ada language constructs at a finer level of granularity, while the AES provides for the evaluation of additional factors and additional tools. The benefit to be derived from using both test suites is the ability to accomplish both objectives. Of course, there may be some users who have only one objective, but making both test suites available does not require anyone to use both.

4.2 Conclusion 2: There is Benefit in Merging the Two Test Suites

All the following standard benefits of combining two products into one should apply to this case if the merger is performed in a rational way:

a. Maintenance and continued enhancement for one product should be easier than for two.

b. A common user interface would make it easier for the user to learn and use.

c. Duplicate tests could be eliminated to reduce the size of the combined product.

d. Focused user experience on one product leading to suggested improvements which benefit the entire Ada community.

One can also focus on four usability attributes (coverage, selectivity, application metrics, user interface) of these evaluation tools and compare the differences between a merged set of tools and separate use of these tools. Coverage is an important usability attribute which, ideally, would provide tests which expose resource usage, robustness, and limitation characteristics of the software under test. Selectivity refers to the user's ability to select tests and parameters which help answer specific user questions: this is another important usability attribute of an evaluation tool (e.g., how much overhead is associated with using generics and/or task rendezvous?) The availability of application specific metrics is also important because there is wide variability of application performance requirements. Finally, the user interface is an important usability attribute because it determines how much effort a user must expend to achieve some control over the results from the tool (e.g., tailoring a test set and metrics for application requirements). Table 1 is a summary of the characteristics of these merged and separate evaluation tools which leads to the conclusion that the merger provides superior usability characteristics. In addition, it is likely that by combining the outputs from the two test suites into a common database that additional information will be obtainable that was not available from either one separately.

Table 1. Summary of Characteristics

	MERGED	SEPARATE
COVERAGE	The ACEC contains 1069 tests and the AES contains 521. Overlap between the two test suites is minimal (53 tests). However, the total number of tests (1590) can be reduced by removing duplicative tests and by using the AES pre-processor to parameterize ACEC tests which differ only in a parameter value. the largest number of ACEC tests (811) are tests of this type.	Each test suite provides less quality information than a combined suite. For example, a user who is interested in capacity limits and comprehensive tasking tests would have to use both test suites.
SELECTIVITY	User option to run all tests or to tailor for application.	ACEC - user must use an editor to tailor test files, then invoke the pre-processor to execute them in batch mode. AES - use only the pre-processor to automatically generate tailored test programs.
APPLICATION METRICS	User options designed to answer application and environment specific questions.	ACEC - figure of merit computed by comparison with a composite reference model. Not application and environment specific. AES - user specified options [e.g., in a unit of measure such as CPU time, memory used], comparison of overhead metrics for selected language constructs, capacity limit. Is application and environment specific.
USER INTERFACE	- Optional modes [interactive/batch] - Menu-driven customization - One interface to learn	ACEC - batch mode interaction depends on system dependent utilities and user's knowledge of how to use them. Pre-processor limited to selecting data capture options (e.g., execution time, code space]. AES - one user interface for interactive initiation and execution options, menu-driven customization of tests, data capture rates and options.

4.3 Conclusion 3: The AES Test Harness Should be the Basis for the Merge

This conclusion stems from the combination of findings that adding the ACEC tests to the AES would be far less expensive than adding the AES tests to the ACEC. In addition, the AES test harness provides a more flexible capability than the present ACEC preprocessor. Granted, the ACEC approach may be easier to use for some situations, but the AES is judged to be easier to use when the full complement of capabilities is being considered. It is also judged easier to expand into new areas in the future because of both the test harness and the database.

4.4 Conclusion 4: Merging the ACEC Into the AES is the Least Expensive Way to Obtain a Comparable Capability in a Single Product

The cost to add the ACEC tests and Median capability to the AES is considered to be substantially below the cost to produce the combined capability any other way. We estimate that one staff-year of focused effort should be adequate to merge the ACEC with the AES. This work should include deleting redundant tests, converting output generation to AES database updates, and converting input formats including combining similar tests into a single parametric test. Much of the repetitive work is expected to be done using editing macros or some other simple automated method.

4.5 Conclusion 5: The Merged Product Must be Easily Partitionable for Ease of Use by Users Who Are Only Interested in an Subset of the Tests

As indicated in Finding 5, hands-on users may not be interested in those tests developed specifically to support third party evaluations, and tool developers are likely to be interested only in those tests related to the tools they are developing. This means that those users who are only interested in a subset should only have to pay for the part of interest, or even if the entire system is free, should only have to load the part of interest in order to run that part. However, evaluations based upon partitioned test suites may result in isolated and subjective data points. A high incidence of disputes and unfairness claims is to be expected from vendors, especially if the tests are not freely available to them.

5. Recommendations

5.1 Recommendation 1: Make Use of Both Test Suites

Since the ACEC and the AES are more complementary than competitive, each should be used for the functions it performs best. The mechanism for using both test suites has to permit selective execution of tests because not all users will need all tests.

5.2 Recommendation 2: Combine Them Under the AES Test Harness and Database

As in Conclusion 3, this conclusion stems from the combination of findings that adding the ACEC tests to the AES would be far less expensive than adding the AES tests to the ACEC. In addition, the AES test harness provides a more flexible capability than the present ACEC preprocessor. Granted, the ACEC approach may be easier to use for some situations, but the AES is judged to be easier to use when the full complement of capabilities is being considered. It is also judged easier to expand into new areas in the future because of both the test harness and the database. However, neither of these current test suites should continue to be developed on their own without regard to the merged capability.

5.3 Recommendation 3: Negotiate for Joint Distribution and Maintenance

The current restrictions on the distribution of the ACEC and the AES will, in a very short time, negate some of the advantages of combining the two systems. We recommend, therefore, that the US enter into a negotiation with the UK to relax those restrictions and to agree on a joint plan for the long term evolution and maintenance of the combined evaluation technology. Negotiations with the UK should attempt to obtain the right for public release of the AES tests; the ACEC should equally be available. Evaluations conducted by DoD evaluation centers should, as a matter of course, include solicited comments by the respective vendor on the evaluation results. Maximum benefit to the Ada community at large will be achieved if such agreements include commercial use as well as military use.

5.4 Recommendation 4: Establish a DoD Program for Ada Compiler and APSE Evaluation

DoD needs to decide how compiler and environment evaluation technology is to be used to its benefit. We recommend the establishment of at least two centers within each service to act as a testing laboratory and distribution point for testing technology. It is appropriate for some DoD programs to make use of the testing technology in a hands-on way, while others should simply buy reports from a central evaluation service. Under the current terms of the MOU that makes the AES available, that service and all its customers must be within DoD. If those conditions are changed, it may become, in time, more appropriate for DoD to buy evaluation results commercially. In the meantime, Ada-based programs need access to the technology, and the DoD should take steps to make it available to them.

5.5 Recommendation 5: Repeat This Evaluation After Delivery of the Next Version of the ACEC

At least two changes in the findings are likely. First, the degree of overlap is likely to increase because ACEC version 2 may include some tests for tools other than the compiler. Second, the cost of merging the ACEC into the AES will be greater because there will be more tests to merge. It is not likely, however, that these changes will be substantial enough to invalidate any of the conclusions or recommendations.

BIBLIOGRAPHY

AES Documents

AES/1 User Introduction to the Ada Evaluation System, Release 1, Version 1, Issue 2, I. Marshall, 27th September 1988

AES/2 Volume 1, Reference Manual for the Ada Evaluation Compiler Tests, Release 1, Version 1, Issue 2, I. Marshall, 5th December 1988

AES/2 Volume 2, Reference Manual for the Ada Evaluation Compiler Tests, Release 1, Version 1, Issue 2, I. Marshall, 5th December 1988

AES/3 Ada Evaluation System User Manual Parts 0 and I Introduction and General Information, Release 1, Version 1, Issue 2, I. Marshall, 25th November 1988

AES/3 Ada Evaluation System User Manual Part IV Evaluation of the Linker and Loader, Release 1, Version 1, Issue 2, I. Marshall, 30th September 1988

AES/3 Ada Evaluation System User Manual Part V Evaluation of the Symbolic Debugger, Release 1, Version 1, Issue 3, I. Marshall, 25th November 1988

AES/3 Ada Evaluation System User Manual Part VI Evaluation of the Version and Configuration Control System, Release 1, Version 1, Issue 2, I. Marshall, 3rd October 1988

AES/3 Ada Evaluation System User Manual Part VII Evaluation of the Pretty Printer, Release 1, Version 1, Issue 3, I. Marshall, 24th November 1988

AES/3 Ada Evaluation System User Manual Part VIII Evaluation of the Editor, Release 1, Version 1, Issue 2, I. Marshall, 25th October 1988

AES/3 Ada Evaluation System User Manual Part X Evaluation of the Requirements Analyzer, Release 1, Version 1, Issue 2, I. Marshall, 19th September 1988

AES/3 Ada Evaluation System User Manual Part XI Evaluation of the Test Support Tools, Release 1, Version 1, Issue 2, I. Marshall, 24th November 1988

AES/3 Ada Evaluation System User Manual Part XIII Evaluation of the Cross-Reference Analyzer, Release 1, Version 1, Issue 2, I. Marshall, 24th November 1988

AES/3 Ada Evaluation System User Manual Part XIV Evaluation of the Name Expander, Release 1, Version 1, Issue 2, I. Marshall, 5th October 1988

AES/3 Ada Evaluation System User Manual Part XV Evaluation of the Source Generator, Release 1, Version 1, Issue 2, I. Marshall, 3rd October 1988

AES/3 Ada Evaluation System User Manual Part XVI Appendices System, Release 1, Version 1, Issue 2, I. Marshall, 5th October 1988

AES/5 Ada Evaluation Test Harness - VAX/VMS Installation Guide, Release, 1 Version 1, Issue 2, S.D. Bluck, 5th October 1988

ACEC Documents

Ada Compiler Evaluation Capability (ACEC) Version Description Document, AFWAL-TR-88-1093, T. Leavitt, K. Terrell, Boeing Military Airplane, August 1988

Ada Compiler Evaluation Capability (ACEC) Reader's Guide, AFWAL-TR-88-1094, T. Leavitt, K. Terrell, Boeing Military Airplane, August 1988

ACEC Technical Operating Report: User's Guide, AFWAL-TR-88-1095, T. Leavitt, K. Terrell, Boeing Military Airplane, August 1988

SEI Documents

Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0, N. Weiderman, March 1989, CMU/SEI-89-TR-13

Other Documents

B. Wichmann, Letter to Dr. John Solomond dated 18 April 1989.

APPENDIX A: AES/ACEC Test Suite Overlap

AES tests are organized into test groups, each of which is identified by a letter of the alphabet. Some of the groups of tests in general do not duplicate the functionality of ACEC tests. These groups are listed in the following section. The section after that (at the same level) lists the AES test groups which partially duplicate the functionality of ACEC tests. In both sections, the AES tests are identified and the differences with the ACEC tests explained. The names and descriptions of each of the tests is include' because the test descriptions themselves make a case that the extent of overlap between the AES and ACEC test suites is minimal. Of all 311 AES compilation system tests, 258 (83%) do not duplicate ACEC tests and 53 (17%) do duplicate ACEC tests.

Many of the tables in this appendix were automatically constructed from data in the AES test suite. Inconsistencies in spelling, the case of letters, and the use of phrases rather than sentences often reflect the actual menu items in the used in the AES Test Harness. No attempt has been made to standardize the entries in those tables. Some of the menu entries use the word "erroneous" when "illegal" would be more in accord with the usual Ada terminology.

A.1 AES Tests Which Don't Duplicate ACEC Tests

The following AES test groups do not duplicate the functionality of ACEC tests:

Table A-1. AES Tests which do not duplicate ACEC Tests

Group	Test Group Name	Number of Tests in Group
A	Compiler Efficiency	22
B	Compiler Informational.Quality	5
C	Compiler Error Reporting	7
D	Compiler Error Recovery	17
E	Compiler Warning	9
F	Compiler Behavioral	6
G	Compiler Capacity	53
K	MASCOT Tasking	7
M	Storage Management	10
N	Input Output	18
Q	Run-Time Limit	7
R	Implementation Dependency	25
S	Erroneous Execution	13
T	Incorrect Order Dependency	17
U	Link/Load	14
	Total	230

The ACEC tests have time and space performance as their test information domain. The ACEC documentation identifies several areas of compiler test information as being outside the scope of the ACEC effort, in particular, questions about compiler features such as automatic recompilation, the quality of error messages, user friendliness, and diagnostics. The AES test domain includes most of these types of compiler test information. All of these test groups determine information about compiler features that varies from compiler to compiler. In particular, Groups B, C, and E seek information that characterizes the quality of messages emanating from the compiler. Groups B, C, D, and E are very similar in form. Groups D, F, M, N, R, and T seek to answer questions about the particular compiler implementation features which can vary from compiler to compiler. Group G is concerned with measuring a compiler's capacity to handle large numbers of Ada language features. Each of these groups is discussed in greater detail below.

A.1.1 Compiler Efficiency (Group A)

The tests in this group measure the speed at which the compiler compiles legal Ada source code, but does not do so in a language feature-by-feature manner. Instead, global issues are probed. For example, tests AA, AB, AC, AD, and AE measure a mix of Ada language features, and tests TA18-TA22 take advantage of the AES preprocessor to determine the relative speed of the compiler with and without listings and with and without other compiler settings in effect. TA25 measures the effect of simultaneous compilations, which is practical information indeed. Finally, tests TA30-TA32 determine whether the compiler can take advantage of information from previous compiles. Thus, while this group at first appears to duplicate ACEC tests, closer examination shows these tests to be of different character from similar tests in the ACEC.

Table A-2. AES Compiler Efficiency Tests

Name	Description
TA01	Compiling a minimal main procedure
TA02	Compiling generic units
TA03	Compiling WITHed units
TA04	Compiling USEd units
TA05	Compiling large uninitialized arrays
TA06	Compiling large initialized arrays
TA08	Producing error messages
TA09	Compiling overloaded identifiers
TA12	Compiling a large number of strings
TA13	Compiling a large number of enumeration literals
TA14	Compiling identifiers with the same name but different scope
TA15	Compiling subunits
TA16	Compiling local optimizations
TA17	Compiling global optimizations
AA	Code-by-the-yard tests compiled in single-user mode
AB	Code-by-the-yard tests compiled in multi-user mode
AC	Code-by-the-yard tests compiled with syntax-only checking
AD	Code-by-the-yard tests compiled with syntax and semantic checking only
AE	Code-by-the-yard tests compiled in batch mode
TA18	Compiling with debug information
TA19	Compiling when listings produced
TA21	Compiling a null procedure with syntax checking only
TA22	Compiling a null procedure with syntax and semantic checking only
TA25	Multiple simultaneous compilations
TA30	Recompilation where only minor modifications have occurred
TA31	Recompilation where only minor modifications to a withed unit have occurred
TA32	Recompilation when there are no changes to the source

A.1.2 Compiler Informational Quality (Group B)

This group causes compilation of valid Ada code such that as many listings as possible are generated. While the process of generating the listings is automatic, the actual evaluation of the listings is not. None of the AES Group B tests duplicates any ACEC tests.

Table A-3. AES Compiler Informational Quality Tests

Name	Description
TB01	Quality of assembler code l sting, data map, concordance listing and general compiler information
TB02	Quality of compilation and elaboration dependency information
TB03	Quality of resolution overloading information
TB04	Quality of listing of calls to the run-time system
TB05	Quality of source related information
TB06	Quality of information relating to the source of dependent compilation units
TB07	Quality of information relating to the optimization of code
TB08	Further test of the quality of information indicating calls to the run-time system

A.1.3 Compiler Error Reporting (Group C)

Each of the tests in this group causes illegal Ada source code to be compiled. The generation of compiler output is automatic, but the evaluation of the results is not. The ACEC contains no tests which examine the behavior of the compiler when it is presented with illegal Ada source code.

Table A-4. AES Compiler Error Reporting Tests

Name	Description
TC01	Reporting of unresolved overloading, no applicable overloading and type mis-match without overloading
TC02	Reporting of erroneous type definitions and hidden identifiers
TC03	Reporting of common mistakes
TC04	Reporting of illegally specified aggregates, illegal non-conformance, illegal declarations in package specifications and illegal type conversions
TC05	Reporting of declarative errors and error clarity
TC06	Reporting of the omission of the prime in an initialized allocation
TC07	Errors hidden by others occurring later

A.1.4 Compiler Error Recovery (Group D)

Each of the tests in this group causes illegal Ada source code to be compiled. The generation of compiler output is automatic, but the evaluation of the results is not. The ACEC contains no tests which examine the behavior of the compiler when it is presented with illegal Ada source code.

Table A-5. AES Compiler Error Recovery Tests

Name	Description
TD01	Recovery from missing semicolons
TD02	Recovery from missing generic keyword
TD03	Recovery from mis-matched BEGIN and END and from missing keywords
TD04	Check whether semantic analysis occurs when syntax errors are found
TD05	Recovery from illegal assignments and use of '_'
TD06	Recovery from mis-spelled keywords
TD07	Recovery from illegal type declarations and discriminants
TD08	Recovery from using wrong subprogam specification keyword
TD09	Recovery from mis-matched parentheses and quotes
TD10	Recovery from compiling CORAL 66 source and Pascal source
TD11	Recovery from using illegal comments
TD12	Recovery from finding the incorrect order of declarations
TD13	Recovery from missing subprogram and package specifications and the use of a specification where a body is required
TD14	Recovery from the use of keywords as identifiers
TD15	Recovery from the use of anonymous array types in record components
TD16	Recovery from the use of a parenthesised range
TD17	Further tests on the recovery from the use of illegal type declarations

A.1.5 Compiler Warning (Group E)

Each of the tests in this group causes legal but suspect Ada source code to be compiled. The generation of compiler output is automatic, but the evaluation of the results is not. The ACEC contains no corresponding tests.

Table A-6. AES Compiler Warning Tests

Name	Description
TE01	Reporting of unrecognized pragmas, pragmas containing syntax errors and illegally placed pragmas
TE02	Reporting of unset variables
TE03	Reporting of endless loops
TE04	Reporting of exceptions which will be raised at run-time
TE05	Reporting of warnings when errors are present
TE06	Reporting of dead variables and dead code
TE07	Reporting of whether a divide by zero is replaced by code which raises an exception
TE08	Further tests on the reporting of unset variables
TE09	Further tests on the reporting of endless loops

A.1.6 Compiler Behavioral (Group F)

These tests examine the behavior of the compiler when it compiles a file containing more than one compilation unit, some of which are legal and some of which are not legal. No ACEC tests deal with any type of illegal Ada source code conditions.

Table A-7. AES Compiler Behavioral Tests

Name	Description
TF01	Compilation of a file containing three compilation units, the second unit being invalid and not a dependent of the third unit
TF02	Compilation of a file containing two compilation units, the first unit being invalid (but already existing in the Program Library) and a dependent of the third unit
TF03	Compilation of a file containing two compilation units, the first unit being invalid (but already existing in the Program Library), the remainder being valid subunits
TF04	Compilation of a file containing three compilation units, the second unit being an invalid generic package body (but already existing in the Program Library) and being instantiated in the third
TF05	Compilation of a file containing two compilation units, the first unit being invalid (but already existing in the Program Library) and not referenced by the second unit
TF06	Compilation of a file containing three compilation units, the first unit being invalid (but already existing in the Program Library), the remainder being valid task subunits

A.1.7 Compiler Capacity (Group G)

This group consists of several tests of compiler capacity. These tests are made possible by the use of the preprocessor which is at the heart of the AES test harness design.

Capacities are not always determined to the nearest unit, since the cost of compiling a family of large Ada source files may be prohibitive. Sometimes, a binary search method is employed to generate Ada source files which are successively closer (over or under the capacity limit) to the real capacity. It is not always necessary to measure capacity to the nearest unit if knowing that a capacity exceeds a large number is sufficient, as it often is. Since the total space that a compiler has must usually be divided between each individual capacity, each capacity exercised separately is likely to be greater than when the capacities are exercised together. This is the motivation for the "code-by-the-yard" tests found in Groups A and V.

No ACEC tests measure compiler capacity.

Table A-8. AES Capacity Tests

Name	Description
TG01	Number of distinct identifiers
TG02	Depth of static nesting of blocks
TG03	Depth of static nesting of packages
TG04	Depth of static nesting of generics
TG05	Expression complexity
TG06A	Number of enumeration literals for an enumeration type
TG06B	Number of IMAGEs of enumeration literals for an enumeration type
TG07	Number of WITHed units
TG08	Number of USEd units
TG09A	Number of elements in a 1D array
TG09B	Number of elements in a 2D array
TG09C	Number of elements in a 3D array
TG10	Number of elements of an aggregate
TG11	Number of components of a record
TG12A	Number of parameters to a procedure
TG12B	Number of parameters to a function
TG13	Number of parameters to a generic unit
TG14	Number of discriminants for a record
TG15	Number of declarations in a declarative part
TG17	Depth of static nesting of variant parts of a record
TG18	Depth of nesting of aggregates
TG19	Number of case statement alternatives
TG20	Precision of universal integer and universal real arithmetic
TG21	Depth of nesting of mixtures of various constructs
TG23	Number of types declarable
TG24	Number of subprograms allowed in a compilation unit
TG25	Number of packages allowed in a compilation unit
TG26	Number of subunits allowed in a compilation unit
TG27	Number of generics allowed in a compilation unit
TG28	Depth of nesting of subprograms
TG29	Depth of nesting of loops
TG30	Depth of nesting of subunits
TG31	Depth of nesting of accept statements
TG32	Depth of nesting of case statements
TG33	Depth of nesting of if statements
TG34	Number of task entries
TG35	Number of array dimensions
TG36	Number of elsif statements
TG37	Number of select statements
TG38	Number of generic subprogram instantiations in a subprogram
TG38A	Number of generic package instantiations in a subprogram
TG39	Number of characters on a line

Table A-9. AES Capacity Tests (Continued)

Name	Description
TG40	Number of characters in an identifier
TG41A	Number of digits in a universal integer of the form 9999...
TG41B	Number of digits in a universal integer of the form 9999...e9
TG41C	Number of digits in a universal integer of the form 7#6666...#e10
TG41D	Number of digits in a universal real of the form 9.9999...
TG41E	Number of digits in a universal real of the form 9.9999...e9
TG41F	Number of digits in a universal real of the form 7#6.6666...#e10
TG42A	Number of characters in an initialized string object
TG42B	Number of characters assigned to an uninitialized string object
TG43	Number of overloaded identifiers
TG44	Number of constraints on a subtype
TG45	Number of identifiers in an identifier list
TG46A	Number of statically nested renamed exceptions
TG46B	Number of statically nested renamed objects
TG46C	Number of statically nested renamed packages
TG46D	Number of statically nested renamed subprograms
TG47	Number of statically nested object names
TG48	Number of types derived from another type
TG49	Number of exceptions declared
TG50	Number of exception handled
TG51	Number of labels on a statement
TG52	Number of tasks in an abort statement
TG53	Number of compilation units allowed in a file
TG54	Number of errors detectable on a single line
TG55	Number of errors detectable in a compilation unit

A.1.8 MASCOT Tasking (Group K)

These tests are tailored to the MASCOT run-time system. Clearly, no overlap exists with any ACEC tests. One might question why tests were written for a specific target processor when the preprocessor permits the writing of more general tests which can be preprocessed into several tests of many target machines. These tests might be suitable for extension to a whole family of target processors.

Table A-10. AES MASCOT Tasking Tests

Name	Description
TK01	Check that pragma PRIORITY is acted upon
TK02	Determine time for simple rendezvous
TK03	Determine time for rendezvous with guards
TK04	Check that expiry of a delay causes an immediate reschedule
TK05	Determination of time-slicing between equal-priority tasks
TK06	Determination of time taken to call CALENDAR.CLOCK
TK07	Determination of whether the occurrence of an interrupt causes an immediate reschedule

A.1.9 Storage Management (Group M)

These tests determine the behaviour of memory management by the compiler run-time system. No ACEC tests exist to determine storage management functions.

Table A-11. AES Storage Management Tests

Name	Description
TM01A	Treatment of STORAGE_ERROR and limits at which it is raised
TM01B	Tests heap followed by stack exhaustion.
TM02A	Same as above
TM02B	Same as above
TM03	Check of UNCHECKED_DEALLOCATION
TM04	Storage reclamation check
TM05	Creeping of heap storage when returning unconstrained types
TM06	Use of STORAGE_SIZE length clause
TM07	Fragmentation of heap storage
TM08	Heap space overhead for allocated objects
TM09	Use of heap storage by the Ada run-time system
TM10	Re-use of heap storage by the Ada run-time system

A.1.10 Input/Output (Group N)

The ACEC Input/Output (I/O) tests determine the speed of GETs and PUTS for reads and writes of different numbers of bytes. The AES tests, on the other hand, determine the behavior of I/O where there are implementation differences. For example, there are tests to determine the effect of control characters, whether input output is buffered, whether the I/O packages are re-entrant, whether restrictions exist on the character set, and whether file sharing is permitted. None of these tests is duplicated by the ACEC.

Table A-12. AES Input Output Tests

Name	Description
TN01	Check whether file deletion is supported
TN02	Check whether file resetting is supported
TN03	Determine the maximum number of open files
TN04	Check whether external file sharing is supported
TN05	Check whether an I/O performing task blocks other tasks
TN06	Check whether the I/O packages are reentrant
TN07	Instantiation with unconstrained arrays and variant records
TN08	What happens to external files on completion of main program
TN09	Examination of the effect of I/O for access types
TN10	Size of a file created for direct access
TN11	Check whether there is a check on the element type
TN12	Examination of the effect of I/O of control characters
TN13	Examination of page and line lengths
TN14	Determination of whether I/O is flushed
TN15	Examination of the effect of file creation on existing files
TN16	Restrictions in the character set accepted by TEXT_IO
TN17	Examination of the rounding of real values
TN18	Determination of whether I/O is buffered

A.1.11 Run-Time Limit (Group Q)

These tests are similar to the Group M Storage Management tests. None of these tests is duplicated by ACEC tests either.

Table A-13. AES Run-Time Limit Tests

Name	Description
TQ01	Maximum number of tasks created by a single program
TQ02	Minimum size of the run-time system
TQ03	Minimum size of the run-time system - no I/O
TQ04	Minimum size of the run-time system - with I/O
TQ05	Size of the tasking system
TQ06	Maximum amount of generated data a program may have
TQ07	Maximum amount of code that may be generated in a compilation unit

A.1.12 Implementation Dependency (Group R)

Since implementation dependency tests are out of the scope of the ACEC, none of these tests is duplicated by the ACEC.

Table A-14. AES Implementation Dependency Tests

Name	Description
TR01	Termination of tasks that depend on library packages
TR02	Restrictions on objects for which pragma SHARED is allowed
TR03	Restrictions on representation clauses
TR04	Restrictions on unchecked conversions
TR05	Values of predefined floating point, fixed point types attributes
TR06	Special circumstances in which NUMERIC_ERROR is raised
TR07	Circumstances in which language-defined pragmas are acted upon
TR08	Rounding convention on conversion of a real number
TR09	Find the value of scalar variables when uninitialized.
TR10	Propagation of user-defined exception out of the main program
TR11	Propagation of predefined exception out of the main program
TR12	Test to determine if lexical replacement characters are allowed.
TR13	IMAGE applied to non-graphic character.
TR14	Generic declaration and body have to be in the same compilation.
TR15	Subunits of a generic unit have to be in the same compilation.
TR16	Determine when bodies of generics are actually instantiated.
TR17	Does pragma INLINE create dependencies between compilation units.
TR18	Type conversion of uninitialized scalar subcomponents.
TR19	Do composite types contain any undeclared extra data fields
TR20	Effects of type CALENDAR.TIME on execution
TR21	Requirements on parameters to results from a main program
TR22	Does optimization create compilation units dependencies
TR23	Program outcome affected by optimizations.
TR24	Determine system dependent values
TR25	Effectiveness of time slicing, and effect of pragma SHARED

A.1.13 Erroneous Execution (Group S)

Tests designed to execute with errors are outside the scope of the ACEC, thus none of these tests overlaps with the ACEC.

Table A-15. AES Erroneous Execution Tests

Name	Description
TS01	Evaluating a scalar variable with an undefined value and attempting to apply a predefined operator to variable that has undefined subcomponents
TS02	Assignment to a variable which is a depending on discriminants, which changes value of the discriminant
TS03	The effect of thc program depends on the passing mechanism
TS04	Call a subprogram with an actual parameter which a subcomponent depending on discriminants, its execution changes the value of the discriminant
TS05	Calling a subprogram which is abandoned by exception, where the action of the program depends the final value of one of its parameters
TS06	A subprogram where the actual parameter changes updating the formal, then tries to use the formal
TS07	Calling a subprogram with an undefined parameter returning an undefined value
TS08	Using value of deferred constant before elaboration of the corresponding full declaration
TS09	Violating the assumptions concerning shared variables
TS10	In which an error situation arises in the absence run-time checks suppressed via pragma SUPPRESS
TS11	Using an address clause to achieve overlays of objects
TS12	Examines what happens when one of two variables, both accessing the same object, deallocated and the other is used to access the object
TS13	An UNCHECKED_CONVERSION which violates the guaranteed for objects of the target type

A.1.14 Incorrect Order Dependency (Group T)

As for Group S, the Group T tests are not duplicated by the ACEC.

Table A-16. AES Incorrect Order Dependency Tests

Name	Description
TT01	Depending on the order of evaluation of default expressions for components or discriminants
TT02	Depending on the order of evaluation of the expressions for the bounds of a range constraint
TT03	Depending on the order of evaluation of the discrete ranges the index constraint of a constrained array definition
TT04	In a constrained array definition, depending on the order of elaboration of the component subtype indication for evaluation of range of index constrain
TT05	In the elaboration of a discriminant constraint, depending on the evaluation order of expressions given in discriminant associations
TT06	For evaluation of an indexed component, depending on the evaluation order of the prefix and the component expressions
TT07	Depending on the order of evaluation of the prefix and discrete range of a slice
TT08	Depending on the order of evaluation of the expressions given in the component associations of an aggregate
TT09	Depending on the order of evaluation of the choices choices of an array aggregate that is not a subaggregate, and the choices of its subaggregates
TT10	Depending on the order of evaluation of the expressions of the component associations of an array aggregate
TT11	Depending on order of evaluation of the operands of either a factor, term, simple expression, relation or expression operands without short circuit form
TT12	Depending on the order of evaluation of the variable name and expression of an assignment statement
TT13	Depending on the order of evaluation of parameter associations of a subprogram call
TT14	Depending on the order of evaluation of any conditions specified in a select alternative
TT15	Depending on the order of evaluation of the task names in an abort statement
TT16	For elaboration of a generic instantiation, depends on evaluation order of each expression supplied as an explicit generic actual parameter
TT17	Test to determine the action taken when there is a dependency on the order of elaboration of the bounds of an array

A.1.15 Link/Load Tests (Group U)

Group U tests the linker and loader. None of the ACEC tests address the linker or loader, thus none of these tests duplicate ACEC tests.

Table A-17. Link/Load Tests

Name	Description
TU01	Test detection of circular elaboration order
TU02	Test detection of missing CUs
TU03	Test detection of obsolete units
TU04	Errors in linking separately compiled subunits
TU05	Errors in linking non-Ada code
TU06	Linking with number of subprograms up to compiler limit
TU07	Errors in linking with generic units
TU08	Errors in linking run-time library components
TU09	Linking large systems
TU10	Linking a unit with same name length as compiler limit
TU11	Test determining maximum number of names in a program
TU12	Test examining overheads of subunits on linking
TU13	Test examining partial linking
TU14	Test examining linking of foreign units

A.2 AES Tests Which Partially Duplicate ACEC Tests

The following groups contain both tests which determine the same information as some ACEC tests and tests which determine different information. Tests which exist in both test suites can have joint value. In some cases, tests from the AES test suite can validate or verify the correct operation and timing of the corresponding test from the ACEC test suite and vice versa.

Table A-18. AES Tests which partially duplicate ACEC Tests

Group	Test Group Name	Number of Tests	Duplicate Tests
I	Compiler Run-Time Efficiency	19	19
J	NPL Test Suite	18	4
L	General Tasking	17	9
O	Optimizing Tests	20	20
V	Dhrystone and code-by-the-yard	7	1
	Total	81	53

A.2.1 Compiler Run-Time Efficiency (Group I)

This group comes closest to the ACEC tests. Most or all of these tests have equivalents in the ACEC.

Table A-19. AES Run-Time Efficiency Tests

Name	Description
TI01A	Efficiency of selecting record components
TI01B	Efficiency of selecting a record within a record
TI01C	Efficiency of making record assignments
TI01D	Efficiency of making record comparisons
TI02A	Efficiency of indexing array components
TI02B	Efficiency of making array assignments
TI02C	Efficiency of making array comparisons
TI02D	Efficiency of using boolean arrays
TI02E	Efficiency of array concatenation
TI02F	Efficiency of array slicing
TI03	Efficiency of matrix operations
TI04	Efficiency of integer computations
TI05	Efficiency of floating point computations
TI05B	Further tests on the efficiency of floating point computations
TI06	Efficiency of fixed point computations
TI07	Efficiency of heap objects
TI08	Efficiency of stack objects
TI09A	Efficiency of generics with parameters of enumerated types
TI09B	Efficiency of generics with parameters of array types
TI09C	Efficiency of generics with parameters of fixed point types
TI09D	Efficiency of generics with parameters of floating point types
TI09E	Efficiency of generics with parameters of record types
TI09F	Efficiency of generics with parameters of discriminated record types
TI09G	Efficiency of generics with subprogram calls
TI10	Efficiency of subprogram calls
TI11	Efficiency of loop statements
TI12	Efficiency of exception handling
TI13	Efficiency of constraint checking
TI14	Efficiency of I/O of scalar types
TI15	Efficiency of I/O of array types
TI16	Efficiency of I/O of record types
TI17	Efficiency of file management operations
TI18	Efficiency of type conversions
TI19	Efficiency of pragma INTERFACE calls

A.2.2 NPL Test Suite (Group J)

Group J contains 18 of the 21 National Physical Laboratory (NPL) tests. That test suite contains some of the same tests or benchmarks that the ACEC carries. In particular, the Gamm, Ackermann, Habermann-Nassi and Whetstone optimization tests are duplicative. The rest of the AES tests in this group appear to not duplicate the ACEC tests.

Table A-20. AES NPL Test Suite Tests

Name	Description
TJ01	Standard Gamm benchmark
TJ02	Standard Whetstone benchmark
TJ03	Standard Ackermann benchmark
TJ04	Formal parameter modes
TJ05	Overloading operators
TJ06	Inline expansion
TJ07	Generics
TJ08	Record types
TJ09	Discriminant types
TJ10	Operator and expression evaluation
TJ11	If statements
TJ12	Task Activation
TJ13	Habermann-Nassi Optimization
TJ14	Subtype declarations
TJ16	Suppressing checks
TJ17	Integer operations
TJ18	Operations of array types
TJ19	Assignment statements

A.2.3 General Tasking (Group L)

The ACEC test suite has approximately 80 tasking tests. Many of the tests in this AES test group are redundant with ACEC tests: tests 1-6, inclusive and tests 15-17, inclusive. The remaining tests appear to be different.

Table A-21. AES General Tasking Tests

Name	Description
TL01	Overhead of task creation
TL02	Effect of idle tasks on performance
TL03	Effect of number of select statements on performance
TL04	Effect of guards on entry statements on performance
TL05	Effect of passing parameters in rendezvous on performance
TL06	Difference in efficiency of having lots of little tasks with single entry choices versus a few big tasks with many select choices
TL07	Effect of ordering on entry clauses in a select
TL08	Check on number of times an else alternative of a selective wait is executed before a reschedule is forced
TL09	Determination of the residual storage of a terminated task
TL10	A check that delay statements are meaningful
TL11	Determination of the overhead of nested accept statements
TL12	Determination of the rules for selecting open accept alternatives
TL13	Determination of the rules for selecting open delay alternatives
TL14	Determination of the overheads involved in processing an interrupt
TL15	Effect of passing various numbers of parameters in rendezvous on performance
TL16	Overheads of conditional entry call and selective wait
TL17	Efficiency of entry families

A.2.4 Optimizing Tests (Group O)

The ACEC contains many tests of optimization and this group appears to duplicate those ACEC optimization tests.

Table A-22. AES Optimizing Tests

Name	Description
TO01	Value propagation
TO02	Common subexpression elimination
TO03	Loop optimizations
TO04	Use of registers for variables/ register allocation
TO05	Inlining subprograms
TO06	Packing data
TO07	Suppressing run-time checks
TO08	Loading only referenced subprograms
TO09	Sharing generic bodies
TO10	Subexpression evaluation
TO11	Further tests on suppressing run-time checks
TO12	Further tests on register allocation
TO13	Loading only referenced subunits
TO14	Removing redundant/unreachable code
TO15	Use of special hardware instructions
TO16	Replacing code by exception raising code
TO17A	"Case" optimizations with an ordered contiguous range
TO17B	"Case" optimizations with a disordered contiguous range
TO17C	"Case" optimizations with an ordered contiguous set of ordered contiguous ranges
TO17D	"Case" optimizations with a sparse random range
TO17E	"Case" optimizations with a dense random range
TO17F	"Case" optimizations with few explicit choices and most of alternatives in 'others'
TO18	Reducing context switching when an accept statement has a null body
TO19	Optimizing a passive task that protects a shared variable
TO20	Optimizing a passive task that controls a buffered channel

A.2.5 Dhrystone and code-by-the-yard (Group V)

The Dhyrstone test is duplicated in the ACEC test suite, but the code-by-the-yard tests are not.

Table A-23. AES Dhrystone and code-by-the-yard Tests

Name	Description
TV01	Dhrystone tests
TV02	Link time of a 21 compilation unit system contained in 1 file. (Executable benchmark, 12,500 LOC)
TV03	Link time of a 9 compilation unit system contained 1 file. 12,500 LOC
TV04	Link time of a 1 compilation unit system contained in 1 file. 12,500 LOC
TV05	Link time of a 35 compilation unit system contained in 3 files. 25,000 LOC
TV06	Link time of a 69 compilation unit system contained in 5 files. 50,000 LOC
TV07	Link time of a 137 compilation unit system contained in 9 files. 100,000 LOC

A.3 Preprocessing Benefits for ACEC Tests

Some of the ACEC tests exhibit a high degree of commonality. The slight differences in each of the test cases may be a result of changing the type of a variable or it may be the result of changing a literal number which must be present in the Ada source and cannot be changed at execution time. In these cases, the AES preprocessor could be used to factor out the changes in a test. Tests so modified would be easier to maintain in the future and the possibility of slight differences in the Ada source code affecting tests results would be reduced. Implementation dependent tests could also benefit from the use of the AES preprocessor, specifically to factor out implementation specific portions of the test programs. Additional tests could be quickly added to the evaluation test suite by factoring out variable types on existing tests. Tests which require large portions of code to be included, such as large exception handling blocks can also benefit. In the table below, ACEC tests are identified which would benefit from being placed into preprocessor form. "Number" refers to the number of tests which could be collapsed to either one or a small number of tests. "Full extent" means that the tests could be collapsed into a single test, "Partial extent" means that some of the tests could be collapsed, but probably into more than one test.

Table A-24. ACEC Tests

Test Name	Description	Number	Extent
delay(n)	Delay Statement	14	full
DES(n)	DES	11	partial
dhry(n)	Dhrystone	3	full
gamm, gamm2	Gamm	2	full
io(n)	I/O/tests	24	partial
reclaim	Reclaim	4	full
Task_num(n)	Tasking	7	full
Task2_num(n)	Tasking	7	full

A.4 ACEC Tests

The following table lists the types of tests found in the ACEC:

Table A-25. ACEC Tests

Number of Tests	Description
811	Language specific tests
12	Avionics application
2	Ackerman's function (classic)
6	Computer Family Architecture (classic)
10	Sort tests (classic)
14	Delay statement tests
11	Data Encryption Standard
3	Dhrystone
1	Electronic Warface application
6	Optimization tests
2	Radar application
2	Gamm (classic)
10	Interrupt handler
24	I/O
1	Kalman filter
25	"Kernal" Livermore loops (classic)
20	Knuth loops (classic)
2	puzzles
4	Reclaim
5	Reed Solomon
1	Runge-Kutta
1	Search
1	Sieve
8	Simulation application
2	Serial Search
1	Procedure call and parameter passing
80	Tasking
4	Whetstone

A.5 Other AES Tests

Table A-26. Other Tests in AES Test Suite

Group	Description	Number
CA	TESTSUITE/CLI/PERFORMANCE	9
CC	TESTSUITE/CLI/ERROR-REP	4
CD	TESTSUITE/CLI/ERROR-RECOVERY	7
CG	TESTSUITE/CLI/CAPACITY	19
CH	TESTSUITE/CHECK-OUT	18
CR	TESTSUITE/CLI/IMPL-DEP	15
CS	TESTSUITE/CLI/SCENARIOS	5
DF	TESTSUITE/DEBUGGER	3
DG	TESTSUITE/DEBUGGER	11
EG	TESTSUITE/EDITOR/CAPACITY	7
ES	TESTSUITE/EDITOR	1
LS	TESTSUITE/PLS/SCENARIO	19
NF	TESTSUITE/NAME EXPANDER	3
NG	TESTSUITE/NAME EXPANDER	5
-	TESTSUITE/PRETTY PRINTER	2
PF	TESTSUITE/PRETTY PRINTER	5
PG	TESTSUITE/PRETTY PRINTER	7
RA	TESTSUITE/RA/PERFORMANCE	4
RG	TESTSUITE/RA/CAPACITY	8
SF	TESTSUITE/SOURCE GENERATOR	2
SG	TESTSUITE/SOURCE GENERATOR	8
TA	TESTSUITE/TST	3
CH	TESTSUITE/CHECK-OUT	6
VG	TESTSUITE/VCCS/CAPACITY	5
VS	TESTSUITE/VCCS/SCENARIO	26
-	TESTSUITE/CROSS REFERENCE ANALYZER	2
XF	TESTSUITE/CROSS REFERENCE ANALYZER	1
XG	TESTSUITE/CROSS REFERENCE ANALYZER	3
ZZ	TESTSUITE/UTILITIES	2
	Total	210

Table A-27. TESTSUITE/CLI/PERFORMANCE

Name	Description
TCA01A	Performance tests for string concatenation operations
TCA01B	Performance tests for string slicing operations
TCA01C	Performance tests for conversion operations
TCA01D	Performance tests for integer arithmetic operations
TCA02	Performance tests for deeply nested conditions
TCA03	Performance tests for FOR-loops
TCA04	Time to enter a command procedure (with and without parameters)
TCA05	Time to enter a command script (with and without parameters)
TCA06	Time to invoke a user-defined tool (with and without parameters

Table A-28. TESTSUITE/CLI/ERROR-REP

Name	Description
TCC01	Error reporting for invoking a non-existent tool
TCC02A	Error reporting for invoking a tool with too few parameters
TCC02B	Error reporting for invoking a tool with the wrong type of parameters
TCC02C	Error reporting for invoking a tool with unknown parameters

Table A-29. TESTSUITE/CLI/ERROR-RECOVERY

Name	Description
TCD01	Error test for using variables of wrong type expressions
TCD02	Error test for using strings in expressions
TCD03	Error test for use of erroneous dereferencing
TCD04	Test for effect of using uninitialized command data
TCD05	Effect of command script termination on files within scripts
TCD06	Error test for specifying parameters more than once
TCD07	Error test for a tool containing an unhandled exception

Table A-30. TESTSUITE/CLI/CAPACITY

Name	Description
TCG01	Maximum number of continuation lines
TCG02A	Maximum size of a command script
TCG02B	Maximum size of a command procedure
TCG02C	Maximum size of a macro declaration
TCG03	Maximum number of variables in a command script
TCG04	Maximum size of string which can be used in a command script
TCG05	Maximum number of arms in a conditional statement in a command script
TCG06A	Maximum depth of nesting of a simple conditional statement in a command script
TCG06B	Maximum depth of nesting of a more general conditional statement in a command script
TCG07	Maximum depth of loop nesting in a command script
TCG08	Maximum depth of command procedure nesting
TCG09	Maximum depth of command script nesting
TCG10	Maximum number of parameters to a command procedure
TCG11	Maximum number of parameters to a command script
TCG12A	Maximum number of elements in a superstring
TCG12B	Maximum level of explicit dereferencing
TCG12C	Maximum number of slices in a string expression
TCG13	Maximum size of arithmetic expression allowed
TCG14	Maximum number of parameters to be passed to a user-defined tool

Table A-31. TESTSUITE/CHECK-OUT

Name	Description
TCH06	Check-out BEGIN, ADA, LINK, EXECUTE and END .PRE
TCH07	Simple test of exception raising and handling
TCH08	Simple test of allocation and deallocation
TCH09	Simple test of tasking features
TCH10	Simple test of passing unconstrained objects
TCH11	Simple test of CALENDAR.CLOCK
TCH01	Configuration and check-out of full RUN_TIME package
TCH02	Checks terminal input and output for executable tests in MANUAL mode
TCH05	Checks support of SEQUENTIAL_IO and DIRECT_IO
TCH12	Simple test of RUN_TIME.START_TEST and RUN_TIME.END_TEST
TCH13	Simple test of RUN_TIME.TIMER
TCH14	Static checkout of the PRETTY JCL-file
TCH15	Static checkout of the XREF JCL-file
TCH16	Static checkout of the EXPAND JCL-file
TCH17	Static checkout of the SOURCEGEN JCL-file
TCH18	Generate RUN_TIME without configuration of LABADR or timing
TCH19	Generate RUN_TIME with timing facilities, but no LABADR
TCH20	Static checkout of the EDITOR JCL-file

Table A-32. TESTSUITE/CLI/IMPL-DEP

Name	Description
TCR01	Examine the use of arithmetic operators + and -
TCR02	Examine the use of arithmetic operators * and /
TCR03	Examine the use of relational operators <, >, = and /=
TCR04	Examine the use of logical operators and / or on relational operators
TCR05A	Examine the use of string concatenation
TCR05B	Examine the use of string subtraction
TCR06	Examine the use of string reduction (slicing)
TCR07	Examine the use of string dereferencing
TCR08	Examine the use of superstrings
TCR09	Examine the use of logical names
TCR10	Examine the use of string to integer conversion
TCR11	Examine the use of integer to string conversion
TCR12	Examine how to extract substrings from strings
TCR13	Examine the use of length operations on strings
TCR14	Examine how to find the offset of substrings in strings

Table A-33. TESTSUITE/CLI/SCENARIOS

Name	Description
TCS05	Test of file searching
TCS06A	Test to execute an Ada program in various types of process
TCS07A	Test to install procedure with no parameters as an APSE tool
TCS07B	Test to install procedure with parameters as an APSE tool
TCS07C	Test to install a command script as an APSE tool

Table A-34. TESTSUITE/DEBUGGER

Name	Description
TDF01	Examine most of the debugger features and behaviour
TDF02	Delay statement handling
TDF03	Examine task execution during debugger input requests

Table A-35. TESTSUITE/DEBUGGER

Name	Description
TDG01	Determine maximum number of break- & watchpoints
TDG02	Determine maximum number of Ada symbols
TDG03A	Processing a file with 250 lines
TDG03B	Processing a file with 500 lines
TDG03C	Processing a file with 1000 lines
TDG03D	Processing a file with 2500 lines
TDG03E	Processing a file with 5000 lines
TDG03F	Processing a file with 10000 lines
TDG03G	Processing a file with 12500 lines
TDG04	Maximum number of tasks that can be monitored
TDG99	Preprocess debug command file

Table A-36. TESTSUITE/EDITOR/CAPACITY

Name	Description
TEG01	Number of distinct identifiers
TEG02	Depth of static nesting of blocks
TEG03	Expression complexity
TEG04	Depth of nesting of mixtures of subprograms, loops, blocks, packages, subunits, accepts, case statements, generics and if statements
TEG05	Number of subprograms allowed in a compilation unit
TEG06	Number of overloaded identifiers
TEG07	Number of compilation units allowed in a file

Table A-37. TESTSUITE/EDITOR

Name	Description
TES01	Generate TES01.TXT and TES02.ADA .. TES04.ADA

Table A-38. TESTSUITE/PLS/SCENARIO

Name	Description
TLS01A	Compile package TVS1 into library A
TLS01B	Compile package TVS1 into sublibrary A1
TLS02A	Compile the TVS2 subsystem into library A
TLS02B	Compile the TVS2 subsystem into library B
TLS02C	Compile the TVS2 subsystem into sublibrary A1
TLS04A	Compile the TVS4 subsystem into library B
TLS04B	Make the TVS4 subsystem obsolete in library B
TLS06A	Compile the TVS6 subsystem into library A
TLS06B	Compile the TVS6 subsystem into library A and time this
TLS06C	Recompile the TVS6 subsystem into library A and time this
TLS10	Compile TLS01 into library A
TLS12A	Obtain time to compile unit TLS03 into empty library A
TLS12B	Obtain time to compile unit TLS03 into full library A
TLS13A	Compile all units of the VCCS Live system into library A
TLS13B	Compile all units of the VCCS Live system into library B
TLS14	Compile TLS04, substitute for TVS1 into library A
TLS99	Preprocess all the sources of the VCCS Live system
TLG01	Examine limits to the depth of dependency structure
TLG02	Examine limits to the number of units automatically recompiled

Table A-39. TESTSUITE/NAME EXPANDER

Name	Description
TNF01	Test containing several types of objects `
TNF02	Test containing long qualification names
TNF03	Compiling the output of test TNF01 and TNF02

Table A-40. TESTSUITE/NAME EXPANDER

Name	Description
TNG01	Number of characters on an input line
TNG02	Number of identifiers
TNG03	Number of USEd units
TNG04	Number of characters in a qualified name
TNG05	Number of identifiers in a qualified name

Table A-41. TESTSUITE/PRETTY PRINTER

Name	Description
PAA	Code-by-the-yard tests pretty printing in single-user mode
PAB	Code-by-the-yard tests pretty printing in multi-user mode

Table A-42. TESTSUITE/PRETTY PRINTER

Name	Description
TPF01	Testing the whole of the ADA syntax
TPF02	Difficult to format source
TPF03	Testing the compilability of pretty printer output
TPF04	Totally unformatted source
TPF99	Set up pretty printer parameters

Table A-43. TESTSUITE/PRETTY PRINTER

Name	Description
TPG01	Number of characters on an input line
TPG02	Depth of nesting of block-structures
TPG03	Depth of nesting of loop-structures
TPG04	Depth of nesting of case-structures
TPG05	Depth of nesting of if-structures
TPG06	Depth of bracketing expressions
TPG99	Set up pretty printer parameters

Table A-44. TESTSUITE/RA/PERFORMANCE

Name	Description
TRA01	Time to enter and leave requirements analyzer with an empty requirements database.
TRA02	Time to enter and leave requirements analyzer with a requirements database holding ten requirements.
TRA03	Time to enter and leave requirements analyzer with a requirements database holding one hundred requirements.
TRA04	Time to enter and leave requirements analyzer with a requirements database holding one thousand requirements.

Table A-45. TESTSUITE/RA/CAPACITY

Name	Description
TRG01	Maximum number of function components in a functional decomposition
TRG02	Maximum number of dataflows in a functional decomposition
TRG03	Maximum number of dataflows that can be connected to a function
TRG04	Maximum number of events definable in total
TRG05	Maximum number of events definable for a function
TRG06	Maximum number of lines of text permitted for the action statements describing a function
TRG07	Maximum number of data type entries in the data dictionary
TRG08	Maximum number of components that may appear in a composite type definition

Table A-46. TESTSUITE/SOURCE GENERATOR

Name	Description
TSF01	Test containing the whole of Ada syntax
TSF02	Test containing complex structures and statements

Table A-47. TESTSUITE/SOURCE GENERATOR

Name	Description
TSG01A	Code-by-the-yard test with 12500 lines
TSG01B	Code-by-the-yard test with 10000 lines
TSG01C	Code-by-the-yard test with 5000 lines
TSG01D	Code-by-the-yard test with 2500 lines
TSG01E	Code-by-the-yard test with 1000 lines
TSG01F	Code-by-the-yard test with 500 lines
TSG01G	Code-by-the-yard test with 250 lines
TSG01H	Code-by-the-yard test with 100 lines

Table A-48. TESTSUITE/TST

Name	Description
TAA	Test Bed Generator Large Sizing Test
TAB	Coverage Analyzer Large Sizing Test
TAC	Timing Analyzer Large Sizing Test

Table A-49. TESTSUITE/CHECK-OUT

Name	Description
TSETUP	Choose timing method and way of obtaining code addresses
TBUILD	(Re-)Build RUN_TIME package
TCH01A	Check-out obtaining code addresses and variable addresses
TCH01B	Check-out target CPU timer and RUN_TIME.EAT
TCH01C	Check-out and determine CPU time for a standard rendezvous
TFINAL	Perform final build of RUN_TIME after all checks ok

Table A-50. TESTSUITE/CROSS REFERENCE ANALYZER

Name	Description
TVG01	Test to find the maximum number of versions of an item...
TVG02	Test to find the maximum number of distinct configurations which may exist.
TVG03	Test to find the maximum number of entities permissible in a configuration.
TVG04	Test to find the maximum depth of hierarchy of configurations which may exist.
TVG05	Test to find the maximum number of configurations in which a component may appear.

Table A-51. TESTSUITE/VCCS/SCENARIO

Name	Description
TVS01A	Preprocessing components so that they can be placed under version control
TVS01B	Placing components for both systems under version control
TVS02A	Building the TVS2 subsystem for the live system
TVS02B	Building the TVS2 subsystem for the training system
TVS02C	Building the TVS3 subsystem for the training system
TVS02D	Building the TVS4 subsystem for the live system
TVS02E	Building the TVS4 subsystem for the training system
TVS02F	Building the TVS5 subsystem for the live system
TVS02G	Building the TVS5 subsystem for the training system
TVS02H	Building the TVS6 subsystem for the live system
TVS02I	Building the TVS6 subsystem for the training system
TVS02J	Building the TVS7 subsystem for the training system
TVS02K	Building the TVS8 subsystem for the live system
TVS02L	Building the TVS8 subsystem for the training system
TVS02M	Building the TVS9 subsystem for the live system
TVS02N	Building the TVS9 subsystem for the training system
TVS03A	Editing files in the TVS2 subsystem for release 2
TVS03B	Editing files in the TVS5 subsystem for release 2
TVS03C	Editing files in the TVS7 subsystem for release 2
TVS03D	Editing files in the TVS9 systems for release 2
TVS03X	This test should only be performed if automatic recompilation is not supported. It recompiles the live system after the amendments made to its components.
TVS03Y	This test should only be performed if automatic recompilation is not supported. It recompiles the training system after the amendments made to its components.
TVS04A	Editing files in the training version of the TVS4 subsystem in release 3
TVS04B	Editing files in the TVS7 subsystem for release 3
TVS04X	This test should only be performed if automatic recompilation is not supported. It recompiles the training system after the amendments made to its components.

Table A-52. TESTSUITE/PRETTY PRINTER

Name	Description
XAA	Code-by-the-yard tests cross referencing in single-user mode
XAB	Code-by-the-yard tests cross referencing in multi-user mode

Table A-53. TESTSUITE/CROSS REFERENCE ANALYZER

Name	Description
TXF01	Several types of objects in several environments

Table A-54. TESTSUITE/CROSS REFERENCE ANALYZER

Name	Description
TXG01	Number of available identifiers
TXG02	Number of references to a single identifier
TXG03	Overall number of references to identifiers

Table A-55. TESTSUITE/UTILITIES

Name	Description
TZZ01	Generate 7 Ada files with 250, ... 12500 lines
TZZ02	Provides a simulated mix of jobs for the TA20 tests

APPENDIX B: Converting ACEC Tests to AES Format

Minimal interfacing to the AES Test Harness (to report success/failure) requires use of the following template:

```
with RUN_TIME;
procedure TA01 is
begin
   RUN_TIME.START_TEST("TA01");
   -- body of test here
   if <test performed ok> then
      RUN_TIME.END_TEST (RUN_TIME.SUCCESS);
   else
      RUN_TIME.END_TEST (RUN_TIME.FAILURE);
   end if;
exception
   when <test specific exception> =>
      RUN_TIME.END_TEST("!<test specific exception>");
{INCLUDE "EXCEPT"}
end TA01;
```

Other results in an ACEC test that would be written to a file and interpreted by a user or by a file formatting program such as ACEC MEDIAN or ACEC FORMAT would instead go to the AES database.

The timing device found in the ACEC include files would have to be replaced also.

The following is one of the ACEC test files as found in the file acker2.a:

```
with text_io ; use text_io;
with calendar;use calendar;
with global;use global;

procedure acker2 is

package int_io is new integer_io ( int );  use int_io;
package flt_io is new float_io ( real );   use flt_io;

pragma suppress(access_check);
pragma suppress(discriminant_check);
pragma suppress(index_check);
pragma suppress(length_check);
pragma suppress(range_check);
pragma suppress(division_check);
pragma suppress(overflow_check);
pragma suppress(elaboration_check);
-----------------------------pragma suppress(storage_check);

m,n,kk:int ;

function ackermann(m,n:in int) return int is
begin
  if    m=0 then return n+1;
  elsif n=0 then return ackermann(m-1,1);
  else           return ackermann(m-1,ackermann(m,n-1));
  end if ;
end ackermann;

begin

pragma include("inittime");
pragma include("startime");
  kk:=ackermann(3,6) ;
pragma include("stoptime0");
  put("acker2 - ackerman(3,6) no pragma suppress ");
pragma include("stoptime2");
--
-- Test Name          :  acker2
-- Prime Purpose      :  Classical test, Ackermann's function, no suppression
--                       intensive test of function calling
-- LRM Features       :  2.4   4.5.2   4.5.3   5.2   5.3   5.8   6.1   6.2   6.3   6.4
--                       6.5
-- LRM Version        :  A
-- Optimization       :
-- Related tests      :  acker1
-- Author             :  Thomas C. Leavitt
-- Reviewer           :
-- Date               :
-- Source             :
-- Dependencies       :
-- Other Information  :
--
  if kk /= 509 then
    put("incorrect result ");
    put(kk);
    new_line;
  end if;

  put("   time per call is");
  put(1000000.0*min_time/172233.0,8,1,0);
  new_line;

end acker2;
```

That file would be enveloped in the AES Test Harness commands until it looked like:

```
with text_io ; use text_io;
with calendar;use calendar;
with global;use global;
with RUN_TIME;      -- added
procedure TA01 is   --"acker2 is " changed to reflect AES test names
package int_io is new integer_io ( int );  use int_io;
package flt_io is new float_io ( real );   use flt_io;

pragma suppress(access_check);
pragma suppress(discriminant_check);
pragma suppress(index_check);
pragma suppress(length_check);
pragma suppress(range_check);
pragma suppress(division_check);
pragma suppress(overflow_check);
pragma suppress(elaboration_check);
-----------------------------pragma suppress(storage_check);

m,n,kk:int ;

TEST_SUCCESS : BOOLEAN;  --  a new boolean to track success/failure
                         --  The new boolean is not always needed.  In
                         --  this case, the expression (kk /= 509)
                   --  already exists and could be used to signal
                   --  to the AES Test Harness whether the test
                   --  succeeded or failed.

function ackermann(m,n:in int) return int is
begin
  if    m=0 then return n+1;
  elsif n=0 then return ackermann(m-1,1);
  else           return ackermann(m-1,ackermann(m,n-1));
  end if ;
end ackermann;

begin
   RUN_TIME.START_TEST("TA01"); -- added to signal start of test
pragma include("inittime");
pragma include("startime");
  kk:=ackermann(3,6) ;
pragma include("stoptime0");
  put("acker2 - ackerman(3,6) no pragma suppress ");
pragma include("stoptime2");
--
-- Test Name         :  acker2
-- Prime Purpose     :  Classical test, Ackermann's function, no suppression
--                      intensive test of function calling
-- LRM Features       :  2.4  4.5.2  4.5.3  5.2  5.3  5.8  6.1  6.2  6.3  6.4
--                       6.5
-- LRM Version        :  A
-- Optimization       :
-- Related tests      :  acker1
-- Author             :  Thomas C. Leavitt
-- Reviewer           :
-- Date               :
-- Source             :
-- Dependencies       :
-- Other Information  :
--
  if kk /= 509 then
    put("incorrect result ");
    put(kk);
    new_line;
    TEST_SUCCESS := false;    -- added
  end if;

  put("   time per call is");
  put(1000000.0*min_time/172233.0,8,1,0);
  new_line;

   if TEST_SUCCESS then                            -- This block added to
      RUN_TIME.END_TEST (RUN_TIME.SUCCESS);        -- record success or
   else                                            -- failure
      RUN_TIME.END_TEST (RUN_TIME.FAILURE);
   end if;
exception
 -- " when <test specific exception> =>" not added as no test specific
 --    "RUN_TIME.END_TEST("!<test specific exception>");" exceptions
 --    occur
{INCLUDE "EXCEPT"} -- added to include exception handlers, doesn't
                   -- have to be added.


end TA01; -- changed from "end acker2;"
```

Distribution List for IDA Paper P-2311

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Dr. John P. Solomond Director Ada Joint Program Office Room 3E114, The Pentagon Washington, D.C. 20301-3081	2
Other	
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
IIT Research Institute 4600 Forbes Blvd., Suite 300 Lanham, MD 20706 Attn. Anne Eustice	1
Mr. Karl H. Shingler Department of the Air Force Software Engineering Institute Joint Program Office (ESD) Carnegie Mellon University Pittsburgh, PA 15213-3890	1
CSED Review Panel	
Dr. Dan Alpert, Director Program in Science, Technology & Society University of Illinois Room 201 912-1/2 West Illinois Street Urbana, Illinois 61801	1
Dr. Thomas C. Brandt 10302 Bluet Terrace Upper Marlboro, MD 20772	1
Dr. Ruth Davis The Pymatuning Group, Inc. 2000 N. 15th Street, Suite 707 Arlington, VA 22201	1

NAME AND ADDRESS	NUMBER OF COPIES
Dr. C.E. Hutchinson, Dean Thayer School of Engineering Dartmouth College Hanover, NH 03755	1
Mr. A.J. Jordano Manager, Systems & Software Engineering Headquarters Federal Systems Division 6600 Rockledge Dr. Bethesda, MD 20817	1
Dr. Ernest W. Kent Philips Laboratories 345 Scarborogh Road Briarcliff Manor, NY 10510	1
Dr. John M. Palms, President Georgia State University University Plaza Atlanta, GA 30303	1
Mr. Keith Uncapher University of Southern California Olin Hall 330A University Park Los Angeles, CA 90089-1454	1
IDA	
General W.Y. Smith, HQ	1
Ms. Ruth L. Greenstein, HQ	1
Mr. Philip L. Major, HQ	1
Dr. Robert E. Roberts, HQ	1
Ms. Anne Douville, CSED	1
Ms. Audrey A. Hook, CSED	2
Dr. Richard L. Ivanetich, CSED	1
Mr. Terry Mayfield, CSED	1
Dr. Richard P. Morton, CSED	2
Ms. Katydean Price, CSED	1
Dr. Richard Wexelblat, CSED	1
Mr. Jonathan D. Wood, CSED	2
IDA Control & Distribution Vault	2